(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

**特開平10－69418**

(43)公開日　平成10年(1998)３月10日

| (51)Int.Cl.[6] | 識別記号 | 庁内整理番号 | ＦＩ | | 技術表示箇所 |
|---|---|---|---|---|---|
| Ｇ０６Ｆ　12/00 | ５３３ | | Ｇ０６Ｆ　12/00 | ５３３Ｊ | |

審査請求　未請求　請求項の数９　ＯＬ　（全 16 頁）

(21)出願番号　特願平9－167571

(22)出願日　平成９年(1997)６月24日

(31)優先権主張番号　９６　１１０　６４９．９

(32)優先日　1996年７月２日

(33)優先権主張国　ドイツ（ＤＥ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

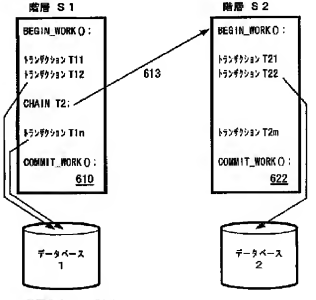
(72)発明者　フランク・ライマン
ドイツ連邦共和国　デー－71134　アイトリンゲン　ハーゼンエッカーヴェーク　19

(74)代理人　弁理士　坂口　博　（外１名）

最終頁に続く

(54)【発明の名称】　階層化トランザクション処理方法

(57)【要約】
【課題】潜在的に分散した多数のトランザクションの総体のアトミック性を保証するのと同時に、前記トランザクションの協調のための通信量ならびに前記トランザクションの総体の並行性挙動および処理量を最適化する。
【解決手段】本発明の根本的な概念は、トランザクションの総体をトランザクションのグループに分割する手法に基づく。各グループは、「トランザクション階層」または単に「階層」と呼ばれる。各階層は、アトミック・コミット・プロトコルによって処理され、同期化されて、それにより、個々の階層ごとにＡＣＩＤ処理結果が保証される。したがって、得られるトランザクションのグループの総体として、すなわち階層の総体として構成される大域トランザクションは、連鎖トランザクション処理の原理にしたがって、連鎖階層のセットとして処理される。連鎖階層実行は、大域トランザクションのアトミック実行を保証する。



1

【特許請求の範囲】
【請求項１】アプリケーション・プログラム定義コミット有効範囲に関してアトミック・コミット・プロトコルによってアプリケーション・プログラムにＡＣＩＤ性質能を提供する少なくとも一つのコンピュータ・システムによるトランザクション処理方法において、
それぞれが少なくとも一つのメンバ・トランザクションを含む、第一のトランザクション階層および第二のトランザクション階層を少なくとも含むことと、
第一のステップにおいて、コミット有効範囲をもつ前記第一のトランザクション階層を処理して、前記アトミック・コミット・プロトコルによって処理結果のＡＣＩＤ性を保証し、さらに、前記第一のステップにおいて、前記第二のトランザクション階層の処理を要求するトランザクション実行要求を待ち行列に挿入することと、
連鎖トランザクションの概念にしたがって、前記第一のステップを、前記第一のトランザクション階層がその処理を正常にコミットする場合にのみ処理される第二のステップと連鎖させることと、
前記第二のステップにおいて、前記待ち行列から前記トランザクション実行要求を取り出すことを含み、さらに、前記第二のステップにおいて、コミット有効範囲をもつ前記第二のトランザクション階層を処理して、前記アトミック・コミット・プロトコルによって処理結果のＡＣＩＤ性を保証することと、を特徴とする方法。
【請求項２】前記待ち行列への前記トランザクション実行要求の挿入が、前記第一のトランザクション階層の前記コミット有効範囲の一部であり、前記待ち行列からの前記トランザクション実行要求の取り出しが、前記第二のトランザクション階層の前記コミット有効範囲の一部である請求項１記載のトランザクション処理方法。
【請求項３】前記待ち行列が多数の異なるコンピュータ・システムに分散している、かつ／または、前記待ち行列に挿入される前記要求がトランザクション保護によって扱われる請求項１または２記載のトランザクション処理方法。
【請求項４】前記待ち行列が、前記トランザクション実行要求を非同期に処理することによって非同期トランザクション処理をサポートしている請求項１～３のいずれか１項記載のトランザクション処理方法。
【請求項５】前記待ち行列が、リソース・マネージャによって管理される耐性保護リソースを表している請求項１～４のいずれか１項記載のトランザクション処理方法。
【請求項６】前記第一のトランザクション階層および／または前記第二のトランザクション階層によって構成される前記メンバ・トランザクションまたは前記メンバ・トランザクションの一部が、何らかの種類のコンピュータ・ネットワークによって接続された多数のコンピュータ・システムによって処理される請求項１～５のいずれか１項記載のトランザクション処理方法。

2

【請求項７】前記アトミック・コミット・プロトコルが２相コミット・プロトコルまたは３相コミット・プロトコルとして実現されている請求項１～６のいずれか１項記載のトランザクション処理方法。
【請求項８】多数の異なるポリシーが、前記階層それぞれの中で処理されるメンバ・トランザクションを定義する請求項１～７のいずれか１項記載のトランザクション処理方法。
【請求項９】局所ノード・ポリシーが、同じトランザクション階層の中で処理すべき、同じコンピュータ・システム上で実行されるすべてのメンバ・トランザクションを定義する請求項８記載のトランザクション処理方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、コンピュータ化トランザクション処理方法に関する。より詳細には、本発明は、トランザクションの総体のアトミック性を保証する、潜在的に分散した多数のトランザクションの総体のコンピュータ化処理に関する。
【０００２】
【従来の技術】トランザクション・アプリケーションとも呼ばれるトランザクションは、回復可能なリソースおよびデータ、たとえばデータベースを一つの整合状態から別の整合状態に変化させる一連の処理を含む。トランザクション処理システムは、あるトランザクションが回復可能なリソースまたはデータに対して何らかの更新を実行し、そのトランザクションがその正常な終了または整合性の中間点に達する前に障害が起こるなら、そのような更新が取り消されること（ロールバック）を保証する。新たな整合性の点に達し、トランザクションによって実施されたすべての更新が永久的にされなければならないとき、トランザクションはコミットする。
【０００３】このトランザクション回復保証を履行するためには、システムは、システム動作不能状態の間、進行中のトランザクションおよびそれらの更新動作の両方を、システムが再起動したとき、回復可能データに対するそれらの効果を正しく反映させることができるよう、記憶することができなければならない。一般に、システムは、トランザクション回復保証を履行するために、ログ・マネージャによって記録され、維持される、各トランザクションの進行状態ならびにリソースおよびデータに対する変更に関するログを維持する。ログ・レコードとして知られるログ・データを審査すると、トランザクションのコミットした動作が、データベースに反映されているか、取り消されたかのいずれかであることを保証することができる。また、ログ・レコードが実データを含むときには、ログ・レコードを利用して、たとえば記憶装置の障害によって損傷または損失したデータを再構成することもできる。ログは、常に成長する順次アクセ

ス・ファイルと考えることができる。

【０００４】ログは、システム障害の際にも無傷で利用可能な状態にとどまる安定な記憶装置、たとえばディスク・ファイル上に永久的に記憶されている。ログ・レコードは、まず、コンピュータのメモリの中の一時的な揮発性ログ・ファイル・バッファに書き込まれたのち、一定の時機（たとえば、トランザクションがコミットされたとき）に安定な記憶装置に移される。

【０００５】ロック・マネージャによってサポートされる「ロッキング」は、共用リソースおよび共用データに対して同時に実行される（すなわち、並行な）トランザクションのアクセスを制御するのに使用され、特に、並行トランザクションが同じリソースおよびデータを不整合に変更してしまうことを防ぐのに使用される。ロッキングを適切に使用すると、トランザクションが操作するリソースおよびトランザクションが読み出すデータが整合状態にあり、リソースおよびデータが他のトランザクションのコミットしていない更新を含まないことをトランザクションに保証することができる。

【０００６】トランザクションが操作する回復可能なリソースおよびデータはさらに、リソース・マネージャによってサポートされることもできる。リソース・マネージャとは、一定のタイプのトランザクション・オブジェクトを管理するサブシステムである。リソース・マネージャは通常、システム全体で利用可能であるかもしれないサービスを、アプリケーションまたは他のリソース・マネージャに提供する。トランザクション・データベース・システム、トランザクション待ち行列マネージャ、トランザクション・セッション・マネージャなどはすべてリソース・マネージャとして働くことができる。

【０００７】トランザクション・マネージャは、コンピュータ・システムの中で並行に処理が起こる多数のトランザクションの流れを統制し、管理し、協調させる。これは、トランザクションのコミットおよび取り消し、すなわちロールバックならびにそれらが失敗したのちのオブジェクト、リソース・マネージャまたはサイトの回復を命令する。

【０００８】局所または分散トランザクション・システム中での一定のトランザクションおよび他すべての並行に処理が起こるトランザクションに関する上述の整合性要件は、ＡＣＩＤ要件としてより正確に表現される。実際、ＡＣＩＤ性とは、以下に説明する４種の異なる部分的要件を組み合わせたものである。

【０００９】・アトミック性（Ａｔｏｍｉｃｉｔｙ）

システム全体の状態に対するトランザクションの変更はアトミック性である。すなわち、すべてが起こるか、何も起こらないかである。これらの変更は、データベース変更、メッセージ、変換器に対する動作などをはじめとする、リソースに対するすべての変更を含む。

【００１０】・整合性（Ｃｏｎｓｉｓｔｅｎｃｙ）

トランザクションは、システム状態の正確な変換である。一つのグループとして受け取られる動作は、状態に関連する整合性制約のいずれをも侵害しない。これには、トランザクションが正しいプログラムであることを要する。

【００１１】・隔離（Ｉｓｏｌａｔｉｏｎ）

トランザクションが並行に実行されるとしても、それぞれのトランザクションＴにとっては、他のトランザクションがＴの前またはＴの後のいずれか（前後両方ではない）で実行されるように見える。

【００１２】・耐性（Ｄｕｒａｂｉｌｉｔｙ）

ひとたびトランザクションが正常に完了し、その作業をコミットするならば、状態に対するその変更は障害に耐えて存続する。すなわち、状態変化は永久的になる。

【００１３】コンピュータ科学の分野では、大域トランザクションのアトミック性を保証する機構が公知である。詳細は、たとえば、「Ｊ．Ｇｒａｙ，Ａ．Ｒｅｕｔｅ，Ｔｒａｎｓａｃｔｉｏｎ Ｐｒｏｃｅｓｓｉｎｇ：Ｃｏｎｃｅｐｔｓ ａｎｄ Ｔｅｃｈｎｉｑｕｅｓ，Ｍｏｒｇａｎ Ｋａｕｆｍａｎｎ Ｐｕｂｌｉｓｈｅｒｓ，Ｓａｎ Ｆｒａｎｃｉｓｃｏ（ＣＡ）」に見いだすことができる。焦点は、トランザクションの総体が整合性のある共通のトランザクション結果、すなわち整合性のシステム状態（すなわちコミットまたはアボートのいずれか）に達することを許す同期プロトコルである、いわゆるアトミック・コミット・プロトコル（ＡＣＰ）にある。もっとも普及し、開発されているＡＣＰは、多数の異なる商業用システムにおいて幾度も実現され、使用されてきた、いわゆる２相コミット（２ＰＣ）プロトコルである。

【００１４】その重要性から、２ＰＣは、企業連合によって標準化された。たとえば、手続き可変要素は、Ｘ／ＯＰＥＮによって標準化された。Ｘ／ＯＰＥＮの詳細は、「Ｘ/ＯＰＥＮ Ｇｕｉｄｅ，Ｄｉｓｔｒｉｂｕｔｅｄ Ｔｒａｎｓａｃｔｉｏｎ Ｐｒｏｃｅｓｓｉｎｇ Ｒｅｆｅｒｅｎｃｅ Ｍｏｄｅｌ（Ｖｅｒｓｉｏｎ ２），Ｘ/ＯＰＥＮ Ｃｏｍｐａｎｙ Ｌｔｄ．，Ｕ.Ｋ．１９９３」に見いだすことができる。オブジェクト指向可変要素は、オブジェクト管理グループ（ＯＭＧ）によって標準化された。オブジェクト管理グループの詳細は、「Ｏｂｊｅｃｔ Ｍａｎａｇｅｍｅｎｔ Ｇｒｏｕｐ，ＯｂｊｅｃｔＴｒａｎｓａｃｔｉｏｎ Ｓｅｒｖｉｃｅｓ（ＯＴＳ），ＯＭＧ Ｄｏｃｕｍｅｎｔ ＴＣ ９４.８.４（１９９４）」に見いだすことができる。対応するＸ／ＯＰＥＮ規格の多くの規格準拠インプリメンテーションが存在し、対応するＯＭＧ規格の多くの準拠インプリメンテーションが進行中であるか、最近リリースされたばかりである。

【００１５】もう一つの公知のＡＣＰは、いわゆる３相コミット・プロトコルである。

【００１６】一つのアトミック作業だけからなる標準的なトランザクションの表現力は非常に限られている。標準的なトランザクションが提供するよりも融通性のある制御手段が求められるであろう多くの状況が存在する。特に実世界の応用の場合には、大域トランザクションと

呼ばれるトランザクションの総体が協力して所望の処理目標を達成しなければならない。

【００１７】そのような複雑なトランザクションのために、種々の異なるモデルが開発されている。例は、セーブポイント技術、入れ子トランザクション、連鎖トランザクションなどである。種々のモデルの概説は、たとえば、「J. Gray, A. Reuter,Transaction Processing: Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco (CA)」に見いだすことができる。

【００１８】これらすべてのトランザクション・モデルは、局所または分散トランザクションの総体、すなわち大域トランザクションのためのアトミック性を保証するための媒体として単独に考慮されるとき、根本的な問題を抱えている。これらの問題に共通の根源は、一方でのアトミック性と他方での並行性との間で適切なバランスを見いだすことの難しさである。一方でアプリケーション全体の整合性を危険にさらさないようにすることは、大域トランザクションの終了までホールドされるロックの数を増やすことを意味する。これは、結果として、アトミック性を保証する協調制御努力の途方もない増大をもたらし、それにより、特に分散処理環境において、性能劣化および並行性における深刻な損失を同時にもたらすであろう。他方、現在の技術水準によると、協調制御努力の減少ならびに性能および並行性における改善は整合性の危険を伴う。

【００１９】大域トランザクションの連鎖トランザクション・モデルは、トランザクションの総体に包含されるトランザクションが最終的に成功することを保証する。この結果は、トランザクションの総体が２ＰＣ処理に包含されると仮定することなく達成され、それにより、上述の問題のいくつかを回避することができる。

【００２０】トランザクション連鎖によると、一つの、すなわち第一のトランザクションをコミットして、もはや必要とされないすべてのオブジェクトを解放し、なおも必要とされる処理コンテキストを、暗黙に開始される次の、すなわち第二のトランザクションに渡すことができる。第一のトランザクションのコミットメントと、次のトランザクションの開始とは、一つのアトミック操作にいっしょに包み込まれる。これは、逆に、他のどのトランザクションも、一方のトランザクションから他方のトランザクションに渡されるコンテキスト・データを見た（または変更した）はずがないことを意味する。連鎖トランザクション・モデルの欠点は、その固有の非同期モードの処理によるものである。第一のトランザクションは、第二のトランザクションの処理の要求を開始するだけであり、第二のトランザクションがすでに始まったかどうかに関して何らかの情報を同期的に戻すこともなく、その処理の結果に関する情報を戻すこともない。連鎖トランザクションのまさにその定義により、このトランザクションは、最終的には、すべての包含されたトランザクションがそれらのサービス要求を受けるということを保証するが、保証された時間枠の中で受けるということは保証できない。したがって、作業単位の概念を提供しながらも、そのような単位はしばしば、ユーザ（エンドユーザとプログラマの両方）がこの作業単位があるべきとみなすものとは対照的である。より理解しやすい（たとえば結合最大期間を有する）作業単位の概念を実現することはやっかいである。

【００２１】

【発明が解決しようとする課題】本発明は、潜在的に分散した多数のトランザクションの総体を処理して、その総体のトランザクション挙動を保証する、すなわちそのアトミック性を保証する課題に基づく。アトミック処理挙動を提供することと同時に、本発明の課題は、前記トランザクションの協調のために通信量を最適化することである。さらに同時に、本発明の課題は、並行性挙動および前記トランザクションの総体の処理能力を最適化することである。

【００２２】

【課題を解決するための手段】本発明の課題は、請求項１によって解決される。大域トランザクションに包含される潜在的に分散したトランザクションの総体を処理するための本発明の根本的な手法は、メンバ・トランザクションの総体をトランザクションのグループに分割する手法に基づく。それぞれのグループは「トランザクション階層」または単に「階層」と呼ばれる。各階層は、アトミック・コミット・プロトコルを介して処理され、同期化され、それにより、個々の階層ごとにＡＣＩＤ処理結果が保証される。したがって、得られたトランザクションのグループの総体として、すなわち階層の総体として構成される大域トランザクションは、連鎖トランザクション処理の原理にしたがって、連鎖階層のセットとして処理される。連鎖階層実行は大域トランザクションのアトミック実行を保証する。

【００２３】本発明によって提案される技法は、トランザクション挙動をメンバ・トランザクションの総体に提供するために本来求められる協調および同期化の努力を有意に減らす。アトミック・コミット・プロトコルによる処理集約的な協調および同期化の努力は、同じ階層に属する比較的少数のメンバ・トランザクション（大域トランザクション全体の一部である潜在的に多数のメンバ・トランザクションに比較して）についてしか求められない。そのような比較的小さなメンバ・トランザクションのグループについてだけ、ＡＣＩＤ性を保証するための処理の負担が求められる。

【００２４】そのうえ、大域トランザクションを多数の階層に分割する結果として、環境全体の並行性挙動および処理能力が増大する。理由は、各階層は、２ＰＣを介して協調された、より少数のトランザクションしか含まず、したがって、階層一つあたりにホールドされるロッ

クがより少ないからである。さらには、各階層は、並行に動作中である他の階層から独立してコミットして、ロックを解除するまでに必要な時間を減らす。換言するならば、要するに、階層化トランザクション処理が粒度を下げ、したがって、並行性およびシステム処理能力を増大させるのである。

【００２５】階層を形成しているメンバ・トランザクションは、それらのＡＣＩＤ性保証結果により、作業の単位とみなすことができるため、階層手法は「作業単位」という新たな概念を実現することができる。たとえば、作業単位は、ユーザとの対話に関して定義することもできる。ユーザの要求によって開始された階層は、コミットしたのち、それによって生じた他の階層の正常な完了を待つ必要なく、ただちに戻ることができる。その結果、ユーザの応答時間は一般に短縮される。

【００２６】連鎖トランザクション処理の概念による階層の連鎖処理により、トランザクションの総体のアトミック性が保証され、したがって、大域トランザクションのトランザクション挙動が提供される。

【００２７】さらなる利点が請求項２によって達成される。本発明のさらなる実施態様によると、第一のトランザクション階層と第二のトランザクション階層とを連鎖するための処理作業は、第一および第二のトランザクション階層の一部である。本発明によると、階層の連鎖は、実行要求を待ち行列に挿入し、それを待ち行列から取り出すことによって達成される。

【００２８】階層の一部としてのこれらの実行要求の扱いは、ＡＣＩＤ操作に参加するための扱いを許し、階層をコミットした時点で、実行要求の扱いが正常に完了することを保証する。

【００２９】さらなる利点が請求項３によって達成される。本発明のこのさらなる実施態様は、非同期トランザクション処理をサポートする待ち行列を教示している。すなわち、トランザクション実行要求を待ち行列に挿入したのち、制御は、要求されたトランザクションがその処理を開始または終了することを待たずに、挿入作業に戻る。

【００３０】非同期トランザクション処理は、特にユーザ対話が関連する場合に多大な応答時間を有意に減らす。

【００３１】さらなる利点が請求項４によって達成される。本発明のさらなる実施態様によると、前記待ち行列は、リソース・マネージャによって管理される耐性保護リソースを表している。

【００３２】保護リソースである待ち行列は、トランザクション実行要求が、ひとたび正常に待ち行列に挿入されると、システムのクラッシュなどの場合でさえ、システムによって常に記憶されることを保証する。最後に、この技術により、一つの階層のアトミック性だけでなく、大域トランザクション全体のアトミック性が保証される。

【００３３】各メンバ・トランザクション、ひいては各階層を任意の巨大で複雑なコンピュータ・ネットワーク内のどこででも処理できるようにすることにより、さらなる利点が達成される。したがって、メンバ・トランザクションおよび階層が多数のコンピュータ・システムの中に分散していてもよい。

【００３４】多数の異なるポリシーが、階層内で処理されるメンバ・トランザクションを定義することを許すことにより、さらなる利点が達成される。したがって、個々の階層をアセンブルするメンバ・トランザクションの選択のための完全な自由が提供される。

【００３５】たとえば、あるポリシーによると、同じコンピュータ・システム上で実行されるメンバ・トランザクションを組み合わせて、個々のトランザクション階層を形成する。このようなポリシーは、潜在的に分散したトランザクションの総体を協調させるのに必要なネットワーク通信量を減らすことができる。一つの同じ物理的機械上で処理されるすべてのメンバ・トランザクションを、必要なトランザクション・マネージャのインスタンスとともに、一つのトランザクション階層の中で処理することができるため、これは、２ＰＣに準拠するために必要なメッセージを機械境界を越えて送ることを回避させ、階層あたりの協調処理の性能を高める。

【００３６】

【発明の実施の形態】本明細書の記載がトランザクションを引用しているならば、局所トランザクションまたは分散トランザクションには限定されない。通常、本明細書の記載によると、トランザクションは、そのようなタイプのいかなるものであってもよい。

【００３７】導入

すでに概説したように、トランザクション・アプリケーションとも呼ばれるトランザクションは、回復可能なリソースおよびデータ、たとえばデータベースを一つの整合状態から別の整合状態に変化させる一連の処理を含む。トランザクション処理システムは、トランザクションが回復可能なリソースおよびデータに対して何らかの更新を実行し、そのトランザクションがその正常な終了または整合性のインテリム・ポイントに達する前に障害が起こるならば、そのような更新が取り消されること（ロールバック）を保証する。新たな整合性の点に達し、トランザクションによって実施されたすべての更新が永久的にされなければならないとき、トランザクションはコミットされる。

【００３８】図１は、疑似コードで表記されたトランザクション・プログラムの全体構造の概要を示す。当然、トランザクション・アプリケーション・プログラムは、いかなる従来の、または特殊なプログラム言語で書くこともできる。このトランザクション・アプリケーション・プログラムは、１行目でBEGIN WORK()を呼び出すこと

により、新たなトランザクションの開始を宣言する。その後、プログラムによって実行されるすべての操作は、このトランザクションの一部になる。また、トランザクション・アプリケーション・プログラムの使用中に他のプログラムによって実行されるすべての操作（２～３行目のプログラム有効範囲の中）もまた、トランザクションの一部になる。プログラムは、図１の５行目のCOMMIT WORK()を呼び出すことにより、トランザクションがシステム状態の完全で正しい変換であることを宣言する。ひとたびトランザクションが正常にコミットすると、トランザクションの効果は耐性かつ持続性になる。トランザクション中に何かがおかしくなるならば、トランザクション・アプリケーションは、７行目でROLLBACK WORK()を呼び出すことにより、すべての操作を取り消すことができる。トランザクションの実行中に障害が出るならば、システムは、トランザクションを一方向的にロールバックさせることができる。そして、BEGIN-COMMITまたはBEGIN-ROLLBACKを使用して、ＡＣＩＤ変換をブラケットすることができる。

【００３９】トランザクション・アプリケーションの基本制御の流れおよびトランザクション処理システム中の主要部品を図２に示す。図２は以下のものを表している。

【００４０】・トランザクション・プログラム２００

【００４１】・潜在的に多数のリソース・マネージャ２１０の一つ。リソース・マネージャとは、一定のタイプのトランザクション・オブジェクトを管理するサブシステムである。リソース・マネージャは通常、システム全体で利用できるかもしれないサービスをアプリケーションまたは他のリソース・マネージャに提供する。

【００４２】・トランザクション・マネージャ２２０。これは、トランザクションのコミットおよびロールバックならびに障害後のオブジェクト、リソース、リソース・マネージャまたは局所もしくは遠隔サイトの回復を命令する。

【００４３】・ログ・マネージャ２３０。これは、障害時にすべてのオブジェクトおよびリソースの整合性のあるバージョンを再構成することができるよう、トランザクションによって実施された変更のログを記録する。

【００４４】・ロック・マネージャ２４０。これは、オブジェクトおよびリソースに対する並行アクセスを規制して、リソース・マネージャがトランザクション隔離を提供するのに役立つ汎用機構を提供する。

【００４５】・トランザクション回復機能２５０。これは、トランザクション・プログラムのコミット有効範囲内で障害の場合に起動される。

【００４６】図２を参照すると、BEGIN WORK()２０１がトランザクションを開始し、それをトランザクション・マネージャに登録し（２０２）、固有のトランザクション識別名２０３を作り出す。ひとたびアプリケーションがトランザクションを開始すると、アプリケーションは、リソース・マネージャ２０４および２０５を呼び出し始め、データベースに対して読み書きを行ったり、局所および遠隔サイトに要求を送付したり、他のタイプのリソースを操作したりすることができる。

【００４７】リソース・マネージャ２１０がそのトランザクションに関連する最初の要求を受け取ると、それはトランザクションに加わり（２１１）、局所トランザクション・マネージャ２２０に対し、それがトランザクションのコミットメントおよびロールバック操作に参与したいことを伝える。いくつかのリソース・マネージャがトランザクションに加わることは通常のことである。これらのリソース・マネージャがトランザクションを代表して作業を実行するとき、リソース・マネージャは、オブジェクトに対して加えた変更のリストを維持する。原則として、リソース・マネージャは、オブジェクトまたはリソースの旧値および新値の両方を記録する。トランザクション処理システムは、そのような変更を記録するためのロギング・サービス２１２を提供する。ログ・マネージャ２３０は、オブジェクトおよびリソースに対するトランザクションのすべての更新の順アクセス・ファイルを効率的に実現する。当然、リソース・マネージャは、これらの更新が何であるかをログ・マネージャに伝えなければならない。

【００４８】隔離を提供するため、リソース・マネージャは、トランザクションによってアクセスされるオブジェクトおよびリソースをロックする（２１３）。これは、他のトランザクションがこのトランザクションのコミットされていない更新を見ることを防ぎ、他のトランザクションが、このコミットされていないトランザクションによって読み書きされるデータを変更することを防ぐ。トランザクション処理システムは、他のリソース・マネージャが使用することができるロック・マネージャ２４０を提供する。

【００４９】トランザクション２００がCOMMIT WORK()２０６を発すると、トランザクション・マネージャは、アトミック・コミット・プロトコル、すなわち、この場合では、２相コミット・プロトコル２６０を実行する。まず、トランザクションに加わったすべてのリソース・マネージャに問い合わせて、トランザクションが整合性のある完全な変換であると思うかどうか尋ねる。いずれかのリソース・マネージャがＮｏと答えると（２６２）、その場合、コミットは失敗する。すべてのリソース・マネージャがＹｅｓと答えるならば（２６２）、トランザクションは正しい変換であり、トランザクション・マネージャはこの事実をログに記録し、各リソース・マネージャに対し、トランザクションが完了したことを知らせる。この時点で、リソース・マネージャは、ロックを解除し、トランザクションを完了するのに必要な他の操作を実行することができる。実行中にトランザク

ョンが失敗するか、あるいは、２ＰＣの相１の間にリソース・マネージャがＮｏと答えるならば、トランザクション・マネージャはトランザクション・ロールバックを命令する。この場合、トランザクション・マネージャは、トランザクションのログを読み、ログ・レコードごとに、そのログ・レコードを書いたリソース・マネージャを呼び出して、そのリソース・マネージャに対し、操作を取り消すよう求める。ひとたび取り消しスキャンが完了すると、トランザクション・マネージャは、トランザクションに加わった各リソース・マネージャを呼び出し、トランザクションがアボートされたことを伝える。

【００５０】トランザクション・マネージャはまた、ノードまたはサイトが障害を起こした場合にもトランザクション回復を命令する。トランザクション・マネージャは、一つのオブジェクトの障害、リソース・マネージャの障害およびサイト全体の障害のための汎用サービスを提供する。

【００５１】以下、トランザクション・マネージャがシステムの回復に役立つ方法を説明する。

【００５２】サイトが障害を起こしたのち、トランザクション処理（ＴＰ）システムがすべてのリソース・マネージャを再起動する。いくつかのトランザクションは障害発生時に進行中であったかもしれない。リソース・マネージャは、それらの再起動ロジックの一部としてトランザクション・マネージャに接続する。そのとき、トランザクション・マネージャは、障害が起きたときに作業中であった各トランザクションの結果をリソース・マネージャに知らせる。いくつかはコミットしていたかもしれず、いくつかはアボートされていたかもしれず、いくつかはコミットする過程にあったかもしれない。リソース・マネージャは、そのコミットした状態を独立して回復することもできるし、トランザクション・マネージャによるログの取り消しおよび繰り返しスキャンに参与することもできる。

【００５３】リソース・マネージャは障害を起こすが、ＴＰシステムの残りが動作を続けるならば、トランザクション・マネージャは、そのリソース・マネージャに関連するすべてのトランザクションをアボートする。リソース・マネージャがサービスに戻ると、トランザクション・マネージャは、それらのトランザクションの結果をリソース・マネージャに知らせる。リソース・マネージャは、この情報およびトランザクション・ログを使用して、その状態を再構成することができる。

【００５４】特定のオブジェクトは損失したが、リソース・マネージャが他の方法で動作しているならば、リソース・マネージャは、他のオブジェクトに関してサービスを提供し続け、その間、失敗したオブジェクトがアーカイブ・コピーおよびそのコピーに対してコミットしたすべての変更のログから再構成される。トランザクション・マネージャおよびログ・マネージャは、アーカイブコピーからのオブジェクトの回復を支援する。

【００５５】各サイトは普通、別個のトランザクション・マネージャを有している。これが、各サイトが互いに独立して動作することを許して、局所自律性を提供する。トランザクションの実行がいくつかのサイトの中に分散しているとき、それは、いくつかのトランザクション・マネージャの中に分散する。その場合、多数のトランザクション・マネージャの中で２ＰＣプロトコルが使用される。

【００５６】リソース・マネージャによって維持、制御される、トランザクション・オブジェクトとも呼ばれるリソースは、保護リソースまたは回復可能リソースである。保護リソースに対して向けられる動作は、それらの動作が完全に終了する前にそれらの結果を外部に出すことはない。これらの更新はコミットメント制御され、正常に終了する前に何かがおかしくなるならばロールバックすることができ、ひとたび正常終了に達すると、一方向のロールバックはなくなる。このように、保護リソースに向けられる動作はＡＣＩＤの性質を有する。

【００５７】図３は、分散ＴＰシステムの側面により焦点を当てたＴＰシステムを示す。さらに、図２に概略したモデルとは異なるいくつかの変更を示す。図３のモデルは、Ｘ／ＯＰＥＮ企業連合の観点のラインに沿っている。

【００５８】図は、コンピュータ・ネットワーク３０３を介して接続された二つのＴＰシステム３０１および３０２を示す。両ＴＰシステムは、それらの間の通信を司る特定のリソース・マネージャ、すなわち通信マネージャ３０４および３０５を含む。さらにこのモデルでは、各リソース・マネージャ３０６および３０７が専用のログおよび専用のロック・マネージャを有するものと仮定する。トランザクションがアボートする場合、リソース・マネージャは、トランザクション・マネージャからの一回のロールバック呼び出しに基づいて、それ自身のロールバックを実行すると考えられる。

【００５９】大域トランザクション

一つのアトミック作業だけからなる標準的なトランザクションの表現力は非常に限られている。標準的なトランザクションが提供するよりも融通性のある制御手段が求められるであろう多くの状況が存在する。特に実世界の応用の場合には、トランザクションの総体が協力して所望の処理目的を達成しなければならない。そのような伝統的なＡＣＩＤトランザクションの総体は、そのアトミック性が保証されるならば、「大域トランザクション」と呼ばれる。

【００６０】コンピュータ科学の分野では、大域トランザクションのアトミック性を保証する機構が公知である。大域トランザクションの問題を扱う一つの手法は、いわゆる「連鎖トランザクション（Chained Transactions）」である。連鎖トランザクションは、ＡＣＩＤ性を保

証するための同期アトミック・コミット・プロトコル（ＡＣＰ）に関する処理努力をある程度減らすための特別な手法を提供する。

【００６１】ＣＴが、標準的なメンバ・トランザクションＴ１～Ｔｎからなる連鎖トランザクションであると仮定する。連鎖トランザクションの概念は、長く続く大域トランザクションＣＴに対して順階階層構造を課すことを許す。この概念は、メンバ・トランザクションＴｉの中から、「ＣＩＡＩＮ」実行要求により、メンバ・トランザクションＴｊのトランザクション処理要求を出すことを許す。二種類の連鎖トランザクションが可能である。同期連鎖トランザクション・モデルによると、メンバ・トランザクションＴｉのコミットメントは、結果的に、連鎖トランザクションＴｊの実行を同期的にもたらす。他方、非同期連鎖トランザクション・モデルでは、メンバ・トランザクションＴｉのコミットメントの後には、連鎖トランザクションＴｊの非同期実行が続く。以下、はっきりと必要ではない限り、連鎖トランザクション原理に関するこれらの異なる表現を区別しない。

【００６２】Ｔｉの処理に拘束された潜在的な実行コンテキスト、たとえばデータベース・カーソルなどを包含するデータベース・コンテキストは、保存し、Ｔｊの処理に渡すことができる。したがって、トランザクション連鎖を用いると、一つのトランザクションＴｉをコミットして、もはや必要ではないすべてのリソースおよびオブジェクトを解放し、なおも必要とされる処理コンテキストを、ＴＰ処理システムによって暗黙に開始される次のトランザクションＴｊに渡すことができる。第一のトランザクションＴｉのコミットメントと、次のトランザクションＴｊの開始とは、一つのアトミック操作にいっしょに包み込まれる。これは逆に、他のトランザクションが、一方のトランザクションＴｉから他方のトランザクションＴｊに渡されるコンテキスト・データを見た、または変更したはずがないことを意味する。

【００６３】連鎖ステップはトランザクション（Ｔｉ）を取り消し不能に完了するため、ロールバックは、連鎖トランザクションＣＴの中の現在作業中のメンバ・トランザクション（ＴＪ）に限定される。Ｔｉのコミットは、トランザクション・アプリケーションＣＴが、その後は必要としないロックを解除することを許す。連鎖トランザクション方式による再起動の扱いは、最新のコミット、すなわち、もっとも最近にコミットされたメンバ・トランザクションのコミットの状態を再び確立することによって実行される。

【００６４】メンバ・トランザクションＴｉの中で次のメンバ・トランザクションＴｊを連鎖するとき、この処理要求が、呼び出し側トランザクションＴｉのコミット有効範囲の一部として、要求待ち行列に挿入される。この要求待ち行列は当然、回復可能、すなわち耐性のリソースであり、この挿入処理はＡＣＩＤ性の要件を満



す。したがって、これらの多数の回復可能な要求待ち行列は、持続性のある回復可能なメッセージングをサポートする。次のメンバ・トランザクションＴｊの処理要求を待ち行列から取り出すことは、Ｔｊのコミット有効範囲の一部である。

【００６５】ＣＴの中でメンバ・トランザクションＴｉから次のメンバ・トランザクションＴｊを連鎖する処理が本質的に非同期的であるならば、これらの非同期トランザクション処理要求は、結果として、次のメンバ・トランザクションＴｊの処理を同時にはもたらさない。その代わりに、処理要求は、呼び出し側トランザクションＴｉのコミット有効範囲の一部として要求待ち行列に挿入される。この処理要求が、結果として、要求された次のメンバ・トランザクションＴｊの実行をいつもたらすかは、ＴＰシステムしだいである。

【００６６】図４は、メンバ・トランザクションＴ１、Ｔ２、Ｔ３、すなわち４０１、４０２、４０３を包含する連鎖トランザクションＣＴの簡単な例を示す図である。そのコミット有効範囲の中で、Ｔ１は、ＡＣＩＤ性が保証される特定の作業を実行する。特定のデータベースのアクセス４０７は一つの例である。同じくそのコミット有効範囲の中で、すなわちその作業をコミットする前に、Ｔ１は、連鎖トランザクションＣＴの一部である次のメンバ・トランザクションＴ２の実行要求を出す（４０４）。ひとたびメンバ・トランザクションＴ２が制御を得ると、それは、同様な方法で、それが可る作業を実行する。特定の時点で、それは、連鎖トランザクションＣＴの一部である最後のメンバ・トランザクションＴ３の実行要求を出す（４０５）。この例によると、メンバ・トランザクションＴ３は、制御を得たのち、そのコミット有効範囲の中でさらなるデータベースにアクセスする（４０６）。

【００６７】本発明はまた、トランザクションがネットワーク中に分散している場合をも取り扱う。そこで、本発明はまた、分散トランザクション処理の分野にも関連する。これは、メンバ・トランザクションそれぞれが、ホストのオペレーティング・システムの別個のスレッドまたはプロセスの中で、またはネットワーク中の異なる物理的構成上で実行するかもしれないことを意味する。したがって、概念的に、一定のトランザクションがネットワーク中のどのコンピュータ・システム上で処理されるのかを区別することはもはや必要ではない。

【００６８】ネットワーク内のコンピュータ・システム間の通信目的のためのメッセージ指向ミドルウェア（ＭＯＭ）の出現により、分散トランザクションの処理のための新たな概念が可能になった。過去数年間、メッセージ指向性ミドルウェア（ＭＯＭ）はその重要性を増した。ＭＱシリーズをもって、ＩＢＭ社は、現在、ＭＱシリーズのプロトコルを主に採用するＩＳＯおよびＯＭＧのような公の標準化機関で進行中の作業によって示さ

れる、この分野における事実上の基準を確立した。ＩＢＭ社のMQシリーズに関しては、たとえば「MQSeries Planning Guide, Document Number GC33-1349-01」を参照されたい。メッセージ指向性ミドルウェアは、リソース・マネージャによって管理される保護リソースの意味での回復可能リソースとして扱われる持続性の回復可能待ち行列を提供する。すなわち、そのような待ち行列の操作をトランザクション内で実施して、メッセージ待ち行列を扱うためのＡＣＩＤ性サポートを保証することができる。

【００６９】図５は、ＭＯＭを利用して、連鎖トランザクション・モデルにしたがって分散大域トランザクションを処理する方法を例示する。ＭＯＭの利用は、二つの連鎖トランザクションの同じ結果を保証するため、非同期プロトコルを考慮している。トランザクション・アプリケーションが二つのメンバ・トランザクションＴ１（５０１）およびＴ３（５０３）に分割され、関連の分散アプリケーションの意味的に正しい実行を結果的にもたらすためには、Ｔ１およびＴ３がいずれも起こらなければならない（すなわち、コミットしなければならない）と仮定する。この場合、Ｔ１は、それがメッセージを出して（５０４）、その局所的な持続性回復可能待ち行列Ｑ１（５０５）の中でＴ３を開始したものと同じ局所トランザクションの中でその対象の回復可能リソースを操作しなければならない。Ｑ１からＱ２（５０６）（Ｔ３に局所的な待ち行列）へのメッセージの伝達は、利用されるメッセージング・ミドルウェア層によって保証され、したがって、第三の独立したトランザクションＴ２（５０２）として認められることができる。Ｔ３は逆に、この要求を、それがその対象の回復可能リソースを操作したものと同じ局所トランザクションの中でその局所待ち行列Ｑ２から得なければならない。その結果、Ｔ１がその作業をコミットするならば、Ｔ３は、最終的に、その関連の作業を正常に実行する。

【００７０】現在の技術水準によって処理される大域トランザクションは、滞在的に分散したトランザクションの総体のアトミック性を保証する根本的な課題を抱えている。

【００７１】たとえば分散トランザクション・アプリケーションは、それぞれがネットワーク中のどこかで実行している二つ以上の意味的に関連した実行スレッドからなることができる。この場合、スレッドの概念は、そのもっとも一般的な意味で、実行中の１片のコードのＯＳ表現として使用され、オペレーティング・システムのディスパッチまたはスワップの単位の厳密な意味では使用されない。このスレッドの総体のサブセットがトランザクションからなるとき、これらのトランザクションの結果は同期化されなければならない。普通、すべての参与するトランザクションがコミットするか、すべてがアボートされるかのいずれかであることが保証されなければならない。２ＰＣは、このアトミック性の課題の解決方法である反面、トランザクションの総体が同じトランザクション結果（すなわち、コミットまたはアボートのいずれか）に達することを許す同期プロトコルであるＡＣＰプロトコルのファミリーの一つの代表として、いくつか周知の欠点を抱えている。たとえば、大域トランザクションの最後まで保持されるロックによるシステム全体の中での並行性の低下や、２ＰＣに固有の、環境全体の性能の低下を招く高いメッセージ複雑性である。

【００７２】これらの欠点は、結果的に、多くの場合、２ＰＣの利用を回避させるインプリメンテーションをもたらす。その結果、アプリケーション全体の整合性が危うくなる。クラッシュの場合には、多くの状況では、やっかいであり、しばしば誤りを起こしやすい手作業によって全体の整合性を再び確立しなければならない。

【００７３】一見すると、連鎖トランザクション処理方式が代替となりうるかもしれない。連鎖トランザクション・モデルは、包含されたトランザクションが最終的には成功するということを保証する。これは、トランザクションの総体が２ＰＣ処理に包含されるものと仮定せずにそれを保証し、したがって、上述の問題を回避させる。

【００７４】連鎖トランザクション・モデルの欠点は、非同期であることに基づく。まさにその定義により、連鎖トランザクション・モデルは、最終的には、すべての包含されたメンバ・トランザクションがそれらのサービス要求を受けることを保証するが、保証された時間枠の中で受けるということは保証できない。したがって、作業単位の概念を提供しながらも、そのような単位はしばしば、ユーザ（エンドユーザとプログラマの両方）がこの作業単位があるべきとみなすものとは対照的である。より理解しやすい（たとえば結合最大期間を有する）作業単位の概念を実現することはやっかいである。

【００７５】そのうえ、連鎖トランザクション・モデルは、完全なＡＣＩＤ機能性を連鎖トランザクションに全体では提供することができないという欠点を抱えている。ＡＣＩＤ性は、個々のメンバ・トランザクションだけに利用できる。

【００７６】階層化大域トランザクション

大域トランザクションに含まれる滞在的に分散したトランザクションの総体を処理するための本発明の根本的な概念は、以下の技術に基づく。

【００７７】・トランザクションの総体をトランザクションのグループに分割する。各グループは「トランザクション階層」または単に「階層」と呼ばれる。

【００７８】・各階層をＡＣＰ（たとえば２ＰＣ）プロトコルによって処理し、同期化し、それにより、個々の階層ごとのＡＣＩＤ処理結果を保証する。

【００７９】・得られるトランザクションのグループの総体として、すなわち階層の総体として構成された大域

トランザクションを、連鎖トランザクションの原理にしたがって、連鎖階層のセットとして処理する。連鎖階層実行は、大域トランザクションのアトミック実行を保証する。

【００８０】階層、すなわちグループに分割されたメンバ・トランザクションの総体を包含する大域トランザクションを処理する手法は、多くの実際の状況で、すべての参加するトランザクションがデフォルトでそれらの終了に同期的に達する必要がないという憶測に基づく。それでも存続することは、すべての参加するトランザクションが同じ判定、すなわちコミットまたはアボートのいずれかに達するということである。この状況では、すべての参加するトランザクションの結果が２ＰＣを介して問用化される必要はない。そのうえ、大域トランザクションを階層の総体に分割する処理は、異なるポリシーにしたがって起こることができる。たとえば、分割方法の例は、ネットワークの同じＴＰシステムで処理されるトランザクションどうしを同じ階層に分類するものでもよい。この手法は、ＡＣＰを介する分散同期化努力を有意に減らす。もう一つの分割ポリシーによると、もっとも重要な回復可能リソースを実施するようなメンバ・トランザクション同士が同じ階層の中で処理される。一般に、参加するトランザクションの総体は、トランザクション結果をＡＣＰを介して同期化しなければならない（たとえば、利用者による作業単位の認識を満たすために）ような、直接協力するトランザクション同士のグループに分割される。利用者の視点からは、階層の構造は見えない。すなわち、それは単に一体のトランザクションとして認識される。同じ物理的機械上で動作する参加するトランザクションのセットまたはエンドユーザによって直接認識される作業単位を実行する参加するトランザクションのセットは当然、階層としての候補である。

【００８１】階層間のトランザクション依存性は、一つの階層の中のトランザクションが一斉にコミットしたならば、他の階層もまたコミットするということである。この状況は、協力するトランザクションのグループ（すなわち階層）が協力するトランザクションの他のグループ（すなわち階層）からサービスを要求する場合に輻輳に見ることができる。各個々の階層のトランザクション結果は、２ＰＣによって管理される。この手法は、各階層の処理に対するＡＣＩＤ性を保証する。階層のセットそのものは、連鎖トランザクション（上記に概説したような連鎖トランザクションの意味で）の総体として管理され、したがって、「非同期トランザクション束」または「連鎖階層」と呼ばれる。非同期トランザクション束を処理するための連鎖手法はアトミック性を保証する。

【００８２】非同期トランザクション束は、そのインプリメンテーションのために、持続性の回復可能メッセージングに依存する。このような持続性の回復可能待ち行列は、ＴＰシステムの一部であるかもしれず、根底のメッセージング・ミドルウェアによって促進されるかもしれない。ＭＱシリーズは、使用することができる持続性の回復可能待ち行列の一例である。このため、非同期トランザクション束に参加する各階層は、その回復可能リソースを操作して、その適切なビジネス機能を、それが独立型のトランザクションであるかのように実行する。これに加えて、階層内の各トランザクションは、根底のメッセージング層の持続性の回復可能待ち行列にメッセージを入れて、包含する分散アプリケーションの意味的成功のために、前者のトランザクションが確かに実行されなければならない、他の階層のトランザクションからのサービスを要求することを可る。アトミック性を達成するためには、これらの要求を待ち行列に入れる動作が適切な各トランザクションの一部として起こるということが決定的である。代わって、同じ理由から、待ち行列中の要求によって開始される階層内の各トランザクションは、要求そのものを取り出すことがその適切なトランザクションの一部であることを確かめなければならない。その結果、階層の中でトランザクションによって実行されるすべての「ＰＵＴ」および「ＧＥＴ」は、包含する階層がコミットする場合にコミットされるか、包含する階層がコミットする場合にしかコミットされない。

【００８３】潜在的に分散したトランザクションの総体を階層に分類し、階層のセットを非同期トランザクションの束として構成すると、得られたトランザクション・モデルを「階層化トランザクション（Stratified Transaction）」と呼ぶ。

【００８４】「ＰＵＴ」および「ＧＥＴ」を適切なトランザクションに参与させるためには、根底の待ち行列インプリメンテーションは、それ自体がリソース・マネージャとして働かなければならない。特に、ＡＣＰ（たとえば２ＰＣ）プロトコルに参加しなければならない。さらには、待ち行列インプリメンテーションの運搬機構は、ひとたび待ち行列システムが「ＰＵＴ」を待ち行列にコミットすると、その送達が保証されなければならないという意味において処理されなければならない。

【００８５】分散トランザクションの総体を協調させるのに要するネットワーク通信量は、２ＰＣに参加するトランザクションを制限して、すべてを一つの同じ物理的機械上で動作させることによって減らすことができる。これは、特定の階層に関連するすべてのトランザクションが（理想的には）同じ機械上で動作するということを意味する。当然、特定の機械またはＴＰシステムが多数の階層のホストであってもよい。連鎖トランザクション・モデルを利用して、階層そのものを参加するトランザクションとして協調させることにより、階層化トランザクション全体の整合性のある結果が達成される。

【００８６】図６は、二つのトランザクション階層Ｓ１（６１０）およびＳ２（６２０）に分割された（何らかのタイプのポリシーにしたがって）簡単な階層化トラン

ザクションＳＴの例を示す。階層Ｓ１は、Ｓ１のコミット有効範囲で処理され、したがって、階層Ｓ１の結果にＡＣＩＤ性を保証するｎ個のメンバ・トランザクションＴ１１（６１１）～Ｔ１ｎ（６１２）を含む。Ｓ１のコミット有効範囲の一部としての階層レベル上の連鎖機構６１３を介して、Ｓ１は、階層Ｓ２の非同期処理を出す。ひとたび階層Ｓ２が制御を取得し、処理を開始すると、そのｍ個のメンバ・トランザクションＴ２１（６２１）～Ｔ２ｍ（６２２）は、Ｓ２のコミット有効範囲で処理されて、ＡＣＩＤ性を保証する。上記の説明によると、メンバ・トランザクションＴ１１～Ｔ１ｎおよび Ｔ２１～Ｔ２ｍのグループの中でだけ、ＡＣＰプロトコルは処理結果を同期化する。

【００８７】図７は、図６の場合を、ただし、個々のトランザクション階層を連鎖するのにＭＯＭを使用する形態で示す。階層Ｓ１の中では、「ＰＵＴ」７０１によって階層Ｓ２の処理要求を挿入することにより、さらなる階層Ｓ２が開始される。分散階層化トランザクションＳＴの意味的に正しい実行をもたらすためには、Ｓ１は、Ｓ１のコミット有効範囲の中でその対象の回復可能リソースを操作しなければならない。したがって、Ｓ２を開始するメッセージをその局所的な持続性の回復可能待ち行列に挿入することは、Ｓ１のコミットの一部である。Ｑ１からＱ２へのメッセージの伝達は、利用されるメッセージング・ミドルウェア層によって保証される。Ｓ２は、逆に、Ｓ２のコミット有効範囲の中でこの要求をその局所待ち行列Ｑ２から「ＧＥＴ」（７０２）しなければならない。

【００８８】最後に、図８は、本発明による、より複雑な階層化トランザクションを示す。この例の大域トランザクションは、４個のトランザクション階層Ｓ１（８１０）、Ｓ２（８２０）、Ｓ３（８３０）、Ｓ４（８４０）に分割されている。各トランザクション階層は、そのコミット有効範囲の中に、メンバ・トランザクションの総体を含む。保護メッセージ待ち行列の中にメッセージを挿入することにより、メンバ・トランザクションＴ１１が階層Ｓ２の処理を要求し（８１１）、メンバ・トランザクションＴ１２が階層Ｓ２およびＳ３の処理を要求し（８１２、８１３）、さらにメンバ・トランザクションＴ１４が階層Ｓ３の処理を要求する。そのうえ、トランザクション階層Ｓ２の処理の間、メンバ・トランザクションＴ２１が階層Ｓ４の処理を要求し（８２１）、メンバ・トランザクションＴ２３が階層Ｓ４の処理を要求する（８２２）。

【００８９】利点

トランザクションの階層化は、潜在的に分散したトランザクションの総体を協調させるのに必要なネットワーク通信量を減らすことができる。これは、種々のポリシーにしたがって、特定の階層に関して含まれるメンバ・トランザクションをグループ分けすることによって達成される。たとえば、一つのグループ分けポリシーによると、一つの同じ物理的機械上で処理されるすべてのメンバ・トランザクションを、必要なトランザクション・マネージャのインスタンスとともに、一つのトランザクション階層の中で処理することもできる。これが、２ＰＣに準拠するために必要なメッセージを機械境界を越えて送ることを回避させ、階層あたりの協調処理の性能を高める。

【００９０】大域トランザクションを多数の階層に分割する結果として、環境全体の並行性挙動および処理能力が増大する。理由は、各階層は、２ＰＣを介して協調された、より少数のトランザクションしか含まず、各階層内のトランザクションの数の減少のため、階層一つあたりにホールドされるロックがより少ないからである。さらには、各階層は、並行に動作中である他の階層から独立してコミットして、ロックを解除するまでに必要な時間を減らす。

【００９１】ユーザの応答時間は一般に短縮される。理由は、ユーザの要求によって開始された階層は、コミットしたのち、それによって生じた他の階層の正常な完了を待つ必要なく、ただちに戻ることができるからである。

【００９２】ネットワーク中心のコンピューティング・パラダイムおよびオブジェクト要求ブローカ・パラダイムがますます採用されると、結果的にトランザクションの拡散が起こる。これらのトランザクションは、アプリケーションに組み合わせなければならない別々のサービスを提供する。この組み合わせは、しばしば、包含されるトランザクションの協調した結果を保証しなければならない。本発明は、現在の技術水準の機構に比較して、概説した利点を有することにより、これを提供する。

【００９３】したがって、意味的な整合性を危うくすることなく、上述のパラダイムの中でアプリケーションの性能を調整するように努力する、分散アプリケーションの実現者は、本教示を利用するための候補者である。

【００９４】略号

２ＰＣ　２相コミット・プロトコル

ＡＣＰ　アトミック・コミット・プロトコル

ＩＳＯ　国際標準化機構

ＭＯＭ　メッセージ指向性ミドルウェア

ＯＭＧ　オブジェクト管理グループ

ＴＭ　トランザクション・マネージャ

ＴＰ　トランザクション処理

【００９５】まとめとして、本発明の構成に関して以下の事項を開示する。

（１）アプリケーション・プログラム定義コミット有効範囲に関してアトミック・コミット・プロトコルによってアプリケーション・プログラムにＡＣＩＤ性機能を提供する少なくとも一つのコンピュータ・システムによるトランザクション処理方法において、それぞれが少なく

とも一つのメンバ・トランザクションを含む、第一のトランザクション階層および第二のトランザクション階層を少なくとも含むことと、第一のステップにおいて、コミット有効範囲をもつ前記第一のトランザクション階層を処理して、前記アトミック・コミット・プロトコルによって処理結果のＡＣＩＤ性を保証し、さらに、前記第一のステップにおいて、前記第二のトランザクション階層の処理を要求するトランザクション実行要求を待ち行列に挿入することと、連鎖トランザクションの概念にしたがって、前記第一のステップを、前記第一のトランザクション階層がその処理を正常にコミットする場合にのみ処理される第二のステップと連鎖させることと、前記第二のステップにおいて、前記待ち行列から前記トランザクション実行要求を取り出すことを含み、さらに、前記第二のステップにおいて、コミット有効範囲をもつ前記第二のトランザクション階層を処理して、前記アトミック・コミット・プロトコルによって処理結果のＡＣＩＤ性を保証することと、を特徴とする方法。

（２）前記待ち行列への前記トランザクション実行要求の挿入が、前記第一のトランザクション階層の前記コミット有効範囲の一部であり、前記待ち行列からの前記トランザクション実行要求の取り出しが、前記第二のトランザクション階層の前記コミット有効範囲の一部である上記（１）記載のトランザクション処理方法。

（３）前記待ち行列が多数の異なるコンピュータ・システムに分散している、かつ／または、前記待ち行列に挿入される前記要求がトランザクション保護によって扱われる上記（１）または（２）記載のトランザクション処理方法。

（４）前記待ち行列が、前記トランザクション実行要求を非同期に処理することによって非同期トランザクション処理をサポートしている上記（１）～（３）のいずれか１項記載のトランザクション処理方法。

（５）前記待ち行列が、リソース・マネージャによって管理される耐性保護リソースを表している上記（１）～（４）のいずれか１項記載のトランザクション処理方法。

（６）前記第一のトランザクション階層および／または前記第二のトランザクション階層によって構成される前記メンバ・トランザクションまたは前記メンバ・トランザクションの一部が、何らかの種類のコンピュータ・ネットワークによって接続された多数のコンピュータ・システムによって処理される上記（１）～（５）のいずれか１項記載のトランザクション処理方法。



（７）前記アトミック・コミット・プロトコルが２相コミット・プロトコルまたは３相コミット・プロトコルとして実現されている上記（１）～（６）のいずれか１項記載のトランザクション処理方法。

（８）多数の異なるポリシーが、前記階層それぞれの中で処理されるメンバ・トランザクションを定義する上記（１）～（７）のいずれか１項記載のトランザクション処理方法。

（９）局所ノード・ポリシーが、同じトランザクション階層の中で処理すべき、同じコンピュータ・システム上で実行されるすべてのメンバ・トランザクションを定義する上記（８）記載のトランザクション処理方法。

【図面の簡単な説明】

【図１】疑似コードで書かれたトランザクション・プログラムの全体構造を示す図である。

【図２】トランザクション・アプリケーションの基本制御の流れおよびトランザクション処理システム内の主要構成部品を示す図である。

【図３】分散トランザクション・モデルを示す図である。

【図４】連鎖トランザクションの簡単な例を示す図である。

【図５】分散大域トランザクションを連鎖トランザクション・モデルによって処理するためにＭＯＭを利用する方法を例示する図である。

【図６】本発明の階層化トランザクションの簡単な例を示す図である。

【図７】個々のトランザクション階層を連鎖するためにＭＯＭを使用する本発明の階層化トランザクションの簡単な例を示す図である。

【図８】多数のトランザクション階層を用いる本発明の階層化トランザクションのより複雑な例を示す図である。

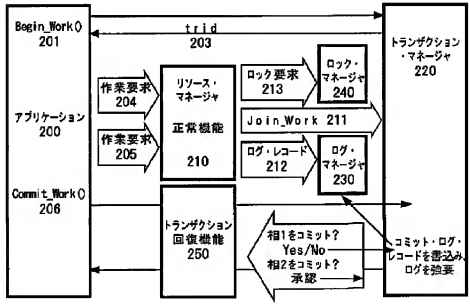
【符号の説明】

| | |
|---|---|
| ２１０、３０６、３０７ | リソース・マネージャ |
| ２２０ | トランザクション・マネージャ |
| ２３０ | ログ・マネージャ |
| ２４０ | ロック・マネージャ |
| ３０４、３０５ | 通信マネージャ |

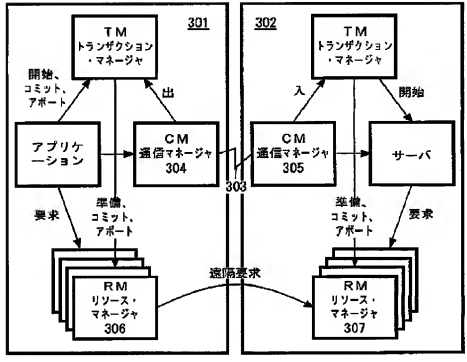
【図１】

```
1 BEGIN_WORK();
2    <any sequence to calls to Resorce Menagers>
3    <any other activity within the commit-scope>
4 if (success)
5    COMMIT_WORK();
6 else
7    ROLLBACK_WORK();
```

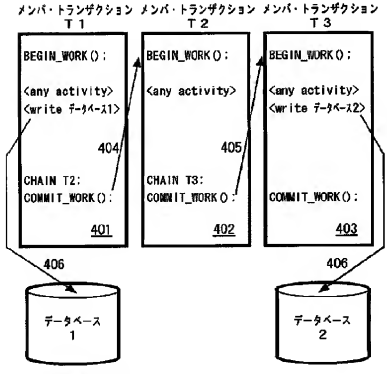
【図２】



【図3】

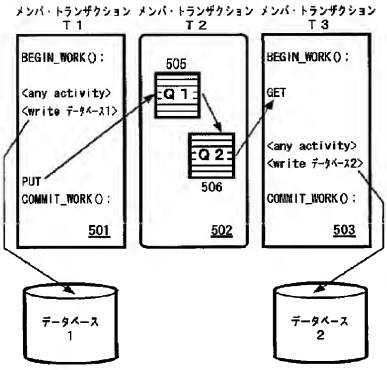



【図4】



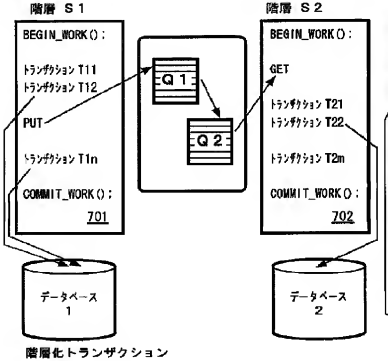

連鎖トランザクション CT＝（T1，T2，T3）

【図5】



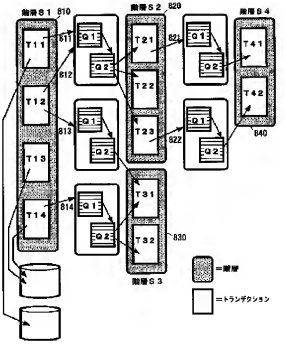

連鎖トランザクション CT＝（T1，T2，T3）

【図7】

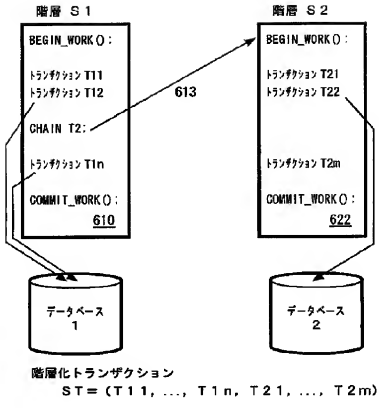



階層化トランザクション
ST＝（T11，...，T1n，T21，...，T2m）

【図8】



【図6】




フロントページの続き

(72)発明者　ディーター・ローラー
ドイツ連邦共和国　デー－71101　シェナ
イヒヘルマン－レンスーヴェーク　5

**PAT-NO:** JP410069418A

**DOCUMENT-IDENTIFIER:** JP 10069418 A

**TITLE:** HIERARCHIZED TRANSACTION PROCESSING METHOD

**PUBN-DATE:** March 10, 1998

**INVENTOR-INFORMATION:**

| NAME | COUNTRY |
|---|---|
| LEYMANN, FRANK DR | |
| ROLLER, DIETER | |

**ASSIGNEE-INFORMATION:**

| NAME | COUNTRY |
|---|---|
| INTERNATL BUSINESS MACH CORP | N/A |

**APPL-NO:** JP09167571

**APPL-DATE:** June 24, 1997

**INT-CL (IPC):** G06F012/00

**ABSTRACT:**

PROBLEM TO BE SOLVED: To assure the total atomicity of many transactions which are latently scattered and also to optimize the traffic necessary for cooperation of those transactions and the total parallel behavior and throughput of the transactions.

SOLUTION: The fundamental concept of this

method is based on a method that divides all transactions into groups, and every group is called 'transaction hierarchy' or just 'hierarchy'. Each of both hierarchies S1 and S2 is processed by an atomic protocol and these hierarchies are synchronized with each other. Thus, the ACID processing result is assured for every hierarchy. Therefore, a global transaction consisting of all groups of transactions to be obtained, i.e., all hierarchies is processed as a set of chained hierarchies based on the chained transaction processing principle. The execution of the chained hierarchies assures the atomic execution of the global transaction.